[研究論文]

逆関数の Taylor 展開

舘野 裕文 [1] ・平山 弘 [2]

1 博士後期課程機械システム工学専攻
2 自動車システム開発工学科

Taylor expansion of inverse function

Hirofumi TATENO [1], Hiroshi HIRAYAMA [2]

Abstract

Though the Taylor series is mathematically basic and important, it has not been used enough in numerical analysis presently. The Taylor series of inverse functions is important to application. In this paper, we compared efficiency of following calculation methods the Picard's method of successive approximation of solving the differential equation of an inverse function, famous the Lagrange inversion formula and the formal Newton's method to inverse functions. We also show that it is possible easily to calculate the maximum and minimum values of the gamma function.

Kewords: taylor expansion, inverse function

1 はじめに

関数を Taylor 展開することは、自動微分 [3] と呼ばれる方法と数学的には同じものである。Taylor 展開した関数は四則演算、微積分、初等関数などの演算を定義することができる。Taylor 級数は係数を浮動小数点で表現することで、高速に Taylor 級数の演算を行うことができる。[5] 倍精度浮動小数点の数値計算で係数に倍精度浮動小数点を使った場合は、係数値に左右されるが数値解析の例題での経験から一般的に 20〜30 次での Taylor 級数の演算が必要であり、本研究ではこの範囲での Taylor 級数を扱う。

Taylor 級数の逆関数計算は方程式の解法などの応用において重要なものである。Taylor 展開された関数の逆関数は、元の関数の 0 次の係数を中心とした Taylor 級数となる。[5] 任意の中心位置での逆関数を求めるためには、求めたい中心位置に近づくように元の関数の中心位置を少しずつずらし、解析接続の繰り返し計算を行う。このため逆関数計算は高速に計算できる必要がある。

本研究では Taylor 級数が与えられる場合の逆関数計算の方法を検討する。逆関数の計算方法として逆関数を満たす微分方程式を Picard の逐次近似法で解く方法、古くから知られている Lagrange inversion formula、形式的 Newton 法を用いて、これらの高速化を行い、計算効率を比較する。300 次程度以上の高次の Taylor 級数の演算の場合は FFT を使うことが有効であるが、本研究では 20〜30 次での Taylor 級数を対象とするため FFT を使った計算は考慮していない。

また Taylor 級数の逆関数計算の応用例として、Gamma 関数の極大極小値を容易に求めることができることを示す。

Taylor 級数の実装には関数及び演算子オーバーロードの可能な Fortran90、C++言語、C#言語が有用である。演算子オーバーロードの使えない言語では、利用方法が煩雑となり、事実上利用不可能になってしまう。本研究では、C++言語を使って実装を行う。また Taylor 級数の係数や中心位置は倍精度浮動小数点を使って表現する。

本研究の実験環境として、CPU は Pentium4 2.20GHz、メモリを 256M で、Red Hat Linux9(Kernel 2.4.20-8) 上において、コンパイラに GCC 3.2.2-5(-O3) を使用して実験を行う。

2 Taylor 級数演算

関数を Taylor 展開するための基本的な考え方を説明し、その計算方法について簡単に述べる。

Taylor 級数の演算は、平行移動によって中心位置を任意の位置へ移すことができる。よって一般性を失うことなしに、任意点を中心とした Taylor 級数を扱うことができる。ここでは原点を中心とした Taylor 級数を考え、次のように定義する。

$$f(x) = f_0 + f_1x + f_2x^2 + f_3x^3 + f_4x^4 \cdots \quad (1)$$
$$g(x) = g_0 + g_1x + g_2x^2 + g_3x^3 + g_4x^4 \cdots \quad (2)$$
$$h(x) = h_0 + h_1x + h_2x^2 + h_3x^3 + h_4x^4 \cdots \quad (3)$$

2.1 Taylor 級数の四則演算

$h(x)$ が $f(x)$ と $g(x)$ の和差積商のとき、$f(x), g(x)$ および $h(x)$ の係数は、それぞれ次のような関係になる。

$$h(x) = f(x) \pm g(x) \quad h_i = f_i \pm g_i \quad (4)$$
$$h(x) = f(x)g(x) \quad h_i = \sum_{j=0}^{i} f_j g_{i-j} \quad (5)$$
$$h(x) = \frac{f(x)}{g(x)}$$
$$h_0 = \frac{f_0}{g_0}, \quad h_i = \frac{1}{g_0}\left(f_i - \sum_{j=0}^{i-1} h_j g_{i-j}\right) \quad (6)$$

2.2 逆数

逆数の係数は次の式によって計算することができる。

$$h(x) = \frac{1}{f(x)}$$
$$h_0 = \frac{1}{f_0}, \quad h_i = -\frac{1}{f_0}\sum_{j=0}^{i-1} h_j f_{i-j} \quad (7)$$

2.3 二乗

Taylor 級数の二乗の計算では次のような関係式が成り立つ。この式によって、二乗の計算は乗算に比べ約二倍の速さで計算することができる。

$$h(x) = (f(x))^2 \quad (8)$$

係数が奇数の場合と偶数の場合に分けて考えれば

$$h_i = \begin{cases} 2\sum_{j=0}^{\frac{i-1}{2}} f_j f_{i-j} & i : odd \\ (f_{\frac{i}{2}})^2 + 2\sum_{j=0}^{\frac{i}{2}-1} f_j f_{i-j} & i : even \end{cases} \quad (9)$$

となり、これらを繰り返し計算する。

2.4 微積分演算

$h(x)$ が $f(x)$ の微積分であるとき、$f(x)$ および $h(x)$ の係数は、それぞれ次のような関係になる。

$$h(x) = \frac{df(x)}{dx} \quad h_i = (i+1)f_{i+1} \quad (10)$$
$$h(x) = \int_0^x f(t)dt \quad h_0 = 0,\ h_i = \frac{1}{i}f_{i-1} \quad (11)$$

2.5 指数関数

指数関数は

$$h(x) = e^{f(x)} \quad (12)$$

とおくと

$$\frac{dh(x)}{dx} = h(x)\frac{df(x)}{dx} \quad (13)$$

を満たす。両辺を Taylor 級数の演算にしたがって解いていき、係数を比較することによって、次のような関係が得られる。

$$h_0 = e^{f_0}, \quad h_i = \frac{1}{i}\sum_{j=1}^{i} j h_{i-j} f_j \quad (14)$$

Taylor 級数の指数関数計算の中には、指数関数の計算は一回しか含まれないので、Taylor 級数の乗算や除算と比べて計算コストはそれほど大きく違わない。このことは、対数関数や三角関数などでも同様である。

2.6 対数関数

対数関数は

$$h(x) = \log f(x) \quad (15)$$

のとき、この関数は次の微分方程式を満たす。

$$f(x)\frac{dh(x)}{dx} = \frac{df(x)}{dx} \quad (16)$$

この式から、指数関数の計算の場合と同様な方法で、次のような関係式が得られる。

$$h_0 = \log f_0,$$
$$h_i = \frac{1}{if_0}\left(if_i - \sum_{j=1}^{i-1} j h_j f_{i-j}\right) \quad (17)$$

2.7 べき乗

べき乗は、α を定数として

$$h(x) = f(x)^\alpha \quad (18)$$

のとき、つぎの微分方程式を満たす。

$$f(x)\frac{dh(x)}{dx} = \alpha h(x)\frac{df(x)}{dx} \quad (19)$$

この式の両辺の係数を比較することで次の関係式が得られる。

$$h_0 = f_0^{\alpha},$$
$$h_i = \frac{1}{i f_0} \sum_{j=1}^{i} \{(\alpha+1)j - i\} f_j h_{i-j} \qquad (20)$$

2.8 三角関数

三角関数

$$g(x) = \sin f(x), \quad h(x) = \cos f(x) \qquad (21)$$

は、次の微分方程式を満たす

$$\frac{dg(x)}{dx} = h(x)\frac{df(x)}{dx}, \quad \frac{dh(x)}{dx} = -g(x)\frac{df(x)}{dx} \qquad (22)$$

この式から、係数に対する次のような関係式が得られる。

$$\begin{aligned} g_0 &= \sin f_0, & h_0 &= \cos f_0 \\ g_i &= \frac{1}{i}\sum_{j=1}^{i} j f_j h_{i-j}, & h_i &= -\frac{1}{i}\sum_{j=1}^{i} j f_j g_{i-j} \end{aligned} \qquad (23)$$

三角関数は、sin と cos を同時に計算すると、計算式が単純で見易い公式となる。これは sinh や cosh の場合も同様である。

3 逆関数計算法

3.1 Picard の逐次近似法

Picard の逐次近似法 [1] を使った逆関数の計算について説明する。関数 f の逆関数を g とすると、$y = f(x)$ は $x = g(y)$ となる。この式を利用し、この両辺を微分すると

$$\frac{dy}{dx} = \frac{1}{f'(y)} = u(y) \qquad (24)$$

という微分方程式を得る。この方程式に $g(x_0) = g_0$ という初期条件を与え、初期値問題にする。これを Picard の逐次近似法を使い、

$$\begin{aligned} g^{\langle 0\rangle}(x) &= g_0, \\ g^{\langle N\rangle}(x) &= g_0 + \int_{x_0}^{x} u\left(g^{\langle N-1\rangle}(t)\right) dt \end{aligned} \qquad (25)$$

で繰り返し演算を行い、逆関数 g の Taylor 級数を求める。式 (25) の $g^{\langle N\rangle}(x)$ の上添え字は反復回数である。

f_c を中心とした n 次までの Taylor 級数 $f(x)$ は

$$f(x) = f_0 + f_1(x - f_c) + f_2(x - f_c)^2 + f_3(x - f_c)^3 + \cdots + f_n(x - f_c)^n \qquad (26)$$

となり、この関数の n 次までの逆関数を求める。式 (25) を Taylor 級数で解くために、$f(x)$ を x 軸方向に $-f_c$、y 軸方向に $-f_0$ 平行移動した関数を $v(x)$ と定義し、初期条件が $v(0) = 0$ になるようにする。これを使って式 (25) を解く。$v(x)$ の中心は原点なので n 次までの Taylor 級数は次のようになる。

$$v(x) = f_1 x + f_2 x^2 + f_3 x^3 + \cdots + f_n x^n \qquad (27)$$

この関数 $v(x)$ の逆関数 $w(x)$ を求め、これを平行移動することで $g(x)$ を求める。ただし、$g(x)$ は f_0 を中心とした Taylor 級数で中心位置を g_c とする。これは

$$\begin{aligned} u(x) &= \frac{1}{v'(x)} \\ &= \frac{1}{f_1 + 2f_2 x + \cdots + n f_n x^{n-1}} \end{aligned} \qquad (28a)$$

$$\begin{aligned} w^{\langle 1\rangle}(x) &= \frac{1}{f_1} x, \\ w^{\langle k\rangle}(x) &= \int_0^x u\left(w^{\langle k-1\rangle}(t)\right) dt \\ & \qquad (k = 2, \cdots, n) \end{aligned} \qquad (28b)$$

$$g_c = f_0, \quad g(x) = f_c + w^{\langle n\rangle}(x) \qquad (28c)$$

となる。Picard の逐次近似法は一回の反復で一つ高次の係数まで正確に求めることができる。逆関数は初期値として一次まで求められるので反復回数は $n-1$ となる。k 回目の反復計算では k 次の係数までは正確に求まるので、$w^{\langle k\rangle}(x)$ は k 次で計算を打ち切ることで計算量を削減できる。また、$w^{\langle k-1\rangle}(t)$ の 0 次の係数は 0 であり、$w^{\langle n\rangle}(x)$ での合成関数計算は $n-1$ 次で打ち切ることから $u(x)$ は $n-1$ 次までの計算で打ち切ることができる。ここまでがほぼ公式通りの計算方法である。

$w^{\langle k\rangle}(x)$ は前ステップの計算において $k-1$ 次まで正確に求まっているので k 次の部分のみを計算すればよい。この最適化を行うと式 (28a)、(28b)、(28c) は

$$\begin{aligned} u_0 &= \frac{1}{f_1}, \quad u_i = -\frac{1}{f_1}\sum_{j=0}^{i-1} u_j f_{i-j+1} \\ & \qquad (i = 1, \cdots, n-1) \end{aligned} \qquad (29a)$$

$$\begin{aligned} w^{\langle 1\rangle}(x) &= \frac{1}{f_1} x, \quad u^{\langle k\rangle}(t) = u\left(w^{\langle k-1\rangle}(t)\right), \\ w^{\langle k\rangle}(x) &= w^{\langle k-1\rangle}(x) + \frac{\left(h^{\langle k\rangle}\right)_{k-1}}{k} x^k \\ & \qquad (k = 2, \cdots, n) \end{aligned} \qquad (29b)$$

$$g_c = f_0, \quad g(x) = f_c + w^{\langle n\rangle}(x) \qquad (29c)$$

となる。ただし、$\left(h^{\langle k\rangle}\right)_{k-1}$ は Taylor 展開した $h^{\langle k\rangle}$ の $k-1$ 次の係数とする。

この式 (29a)、(29b)、(29c) の中で最も計算量が多いのは、式 (29b) の中にある合成関数を計算する部分なので、この部分について計算量の削減を行う。$w^{\langle k-1\rangle}(t)$ の 0 次の係数は 0 なので

$$w^{\langle k-1\rangle}(t) = t\left\{p^{\langle k-1\rangle}(t)\right\} \qquad (30)$$

$$
\begin{array}{c|cccccc}
\left(h^{\langle k\rangle}\right)_1 & \left(p^{\langle k-1\rangle}\right)_0 & & & & & \\
\left(h^{\langle k\rangle}\right)_2 & \left(p^{\langle k-1\rangle}\right)_1 & \left(\left(p^{\langle k-1\rangle}\right)^2\right)_0 & & & & \\
\vdots & \vdots & \vdots & \ddots & & & \\
\left(h^{\langle k\rangle}\right)_{k-2} & \left(p^{\langle k-1\rangle}\right)_{k-3} & \left(\left(p^{\langle k-1\rangle}\right)^2\right)_{k-4} & \cdots & \left(\left(p^{\langle k-1\rangle}\right)^{k-2}\right)_0 & \\
\left(h^{\langle k\rangle}\right)_{k-1} & \left(p^{\langle k-1\rangle}\right)_{k-2} & \left(\left(p^{\langle k-1\rangle}\right)^2\right)_{k-3} & \cdots & \left(\left(p^{\langle k-1\rangle}\right)^{k-2}\right)_1 & \left(\left(p^{\langle k-1\rangle}\right)^{k-1}\right)_0
\end{array}
$$

図 1: Picard の逐次近似法における $h^{\langle k\rangle}$ と $p^{\langle k-1\rangle}$ の関係

とする関数 $p^{\langle k-1\rangle}(t)$ を考えることができる。式 (29b) からわかるように合成関数は $k-1$ 次までの計算になるので $h^{\langle k\rangle}(t)$ は

$$
\begin{aligned}
h^{\langle k\rangle}(t) &= u_0 + \sum_{i=1}^{k-1}\sum_{j=1}^{i} u_j \left(\left(p^{\langle k-1\rangle}\right)^j\right)_{i-j} t^i \\
&= \frac{1}{f_1} + \sum_{i=1}^{k-1}\sum_{j=1}^{i} u_j \left(\left(p^{\langle k-1\rangle}\right)^j\right)_{i-j} t^i
\end{aligned}
\tag{31}
$$

とすることができる。これから $\left(h^{\langle k\rangle}\right)_i$ は

$$
\left(h^{\langle k\rangle}\right)_i = \sum_{j=1}^{i} u_j \left(\left(p^{\langle k-1\rangle}\right)^j\right)_{i-j} \tag{32}
$$

となる。

式 (32) の $\left(h^{\langle k\rangle}\right)_i$ の計算における $\left(p^{\langle k-1\rangle}\right)^j$ の係数を書き並べると図 1 のような関係となる。式 (29b) から反復計算において $\left(h^{\langle k\rangle}\right)_{k-1}$ のみ求めればよいが、$\left(h^{\langle k\rangle}\right)_{k-1}$ での $\left(\left(p^{\langle k-1\rangle}\right)^j\right)_{i-j}$ は、べき乗の式 (20) からわかるように $\left(h^{\langle k\rangle}\right)_{k-2}$ 以前での $\left(\left(p^{\langle k-1\rangle}\right)^j\right)_{i-j}$ の計算値が必要となる。$\left(h^{\langle k\rangle}\right)_{k-1}$ での $\left(\left(p^{\langle k-1\rangle}\right)^j\right)_{i-j}$ は、$\left(h^{\langle k\rangle}\right)_{k-2}$ 以前での $\left(\left(p^{\langle k-1\rangle}\right)^j\right)_{i-j}$ での計算値と一致しているので、この計算値を使いまわせば、計算量を大幅に削減できる。

また、$\left(p^{\langle k-1\rangle}\right)^j$ は連続な自然数でのべき乗なので、べき乗の式を使わずにすでに計算している値を用いて、

$$
\left(p^{\langle k-1\rangle}\right)^j = \begin{cases} \left(p^{\langle k-1\rangle}\right)^{\frac{j}{2}} \left(p^{\langle k-1\rangle}\right)^{\frac{j}{2}} & j : even \\ p^{\langle k-1\rangle} \left(p^{\langle k-1\rangle}\right)^{j-1} & j : odd \end{cases} \tag{33}
$$

という乗算の形に変形し、さらに j が偶数の場合には乗算を行わず二乗の式 (9) を使って計算することで計算量を削減できる。

3.2 Lagrange inversion formula

Lagrange inversion formula [2] は、与えられた関数 f について

$$
v(x) = f(x) - f(0) \tag{34a}
$$

$$
w = \frac{x}{v(x)} \tag{34b}
$$

$$
s_n = \frac{1}{n!}\left\{\left(\frac{d}{dx}\right)^{n-1} w^n\right\}_{x=0} \tag{34c}
$$

$$
g(x) = \sum_{n=1}^{\infty} s_n \left(x - f(0)\right)^n \tag{34d}
$$

として逆関数 g を求める。ただし、$g(x)$ は $f(0)$ での Taylor 級数である。ここで $f(x)$ が式 (26) の形で与えられ、n 次までの逆関数を求めるとすると次のように展開、整理される。

$$
\begin{aligned}
w &= \frac{x}{f(x) - f_0} \\
&= \frac{1}{f_1 + f_2 x + \cdots + f_n x^{n-1}}
\end{aligned}
\tag{35a}
$$

$$
h^{\langle i\rangle} = w^i \qquad (i = 1, \cdots, n) \tag{35b}
$$

$$
\begin{aligned}
g_c &= f_0, \quad g_0 = f_c, \\
g_i &= \frac{\left(h^{\langle i\rangle}\right)_{i-1}}{i} \qquad (i = 1, \cdots, n)
\end{aligned}
\tag{35c}
$$

$h^{\langle n\rangle}$ において $n-1$ 次まで求めるため、べき乗の計算を行う w は $n-1$ 次まで求める必要がある。

この中で式 (35b) のべき乗が最も計算時間を要する部分になる。

このべき乗部分の解き方には二つの方法がある。一つはこのべき乗が連続した自然数によるべき乗であることを利用し、Picard の逐次近似法の場合と同じように式 (33) の乗算の形にして計算中の値を保持しながら解く方法である。乗算の形でべき乗部分を計算する場合は、全てのべき数で $n-1$ 次まで計算する必要がある。これは、乗算において m 次までの厳密な計算をするには掛け合わされる Taylor 級数が共に m 次まで必要となるため、式 (35c) で最も大きいべき数 n での計算に必要な $n-1$ 次の係数に合わせて各べき乗を計算するためである。

もう一つは、べき乗の式 (20) で解く方法である。この方法では式 (35a) と式 (35b) の間で約分をすることで計算量を若干減らすことができる。具体的に書くと式 (35a)、(35b)、(35c) は次のようになる。

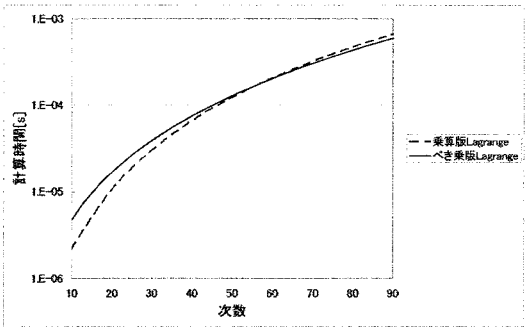

図 2: Lagrange inversion formula

$$w_0 = 1, \quad w_i = -\frac{1}{f_1}\sum_{j=0}^{i-1} w_j f_{i-j+1} \quad (i = 1, \cdots, n-1) \tag{36a}$$

$$h^{\langle 1\rangle} = w, \quad \left(h^{\langle i\rangle}\right)_0 = \left(\frac{1}{f_1}\right)^i,$$
$$\left(h^{\langle i\rangle}\right)_j = \frac{1}{j}\sum_{k=1}^{j}\{(i+1)\,k - j\}\, w_i \left(h^{\langle i\rangle}\right)_{j-k} \quad (j = 1, \cdots, i-1) \quad (i = 2, \cdots, n) \tag{36b}$$

$$g_c = f_0, \quad g_0 = f_c,$$
$$g_i = \frac{\left(h^{\langle i\rangle}\right)_{i-1}}{i} \quad (i = 1, \cdots, n) \tag{36c}$$

べき乗部分を乗算で解いた場合と違いべき乗の式で解いた場合は各べき乗計算における最大計算次数は $i-1$ 次となる。

乗算で解いた場合とべき乗で解いた場合を比較すると図 2 のようになる。

60 次を超えた辺りから乗算よりもべき乗が速く計算できるのは、乗算の場合は計算結果を再利用するために最終的に必要な $n-1$ 次までの計算を始めから行うのに対して、べき乗は計算コストは乗算より高いが $i-1$ 次までの計算ですむ為である。

この結果から 20〜30 次までの範囲では、乗算を使った方法が優れている。

3.3 形式的 Newton 法

関数 f の逆関数は、$x = f(y)$ となるので、

$$x - f(y) = 0 \tag{37}$$

を Newton 法を使って解くことで、逆関数を求めることができる。Newton 法の一般式は

$$Y^{\langle i+1\rangle} = Y^{\langle i\rangle} - \frac{F(Y^{\langle i\rangle})}{F'(Y^{\langle i\rangle})} \tag{38}$$

なので、式 (37) を適応すると、逆関数 g についての反復式は

$$g^{\langle i+1\rangle} = g^{\langle i\rangle} - \frac{f(g^{\langle i\rangle}) - x}{f'(g^{\langle i\rangle})} \tag{39}$$

となる。式 (39) の初期値は任意の値でよい。

$f(x)$ が式 (26) の Taylor 級数の形で与えられられる場合、式 (39) をそのまま使って求めた n 次までの逆関数 $g(x)$ は 0 を中心とした Taylor 級数となる。Taylor 展開された関数 $f(x)$ の逆関数は、f_0 を中心とした Taylor 級数として求まることから、逆関数の展開の中心 f_0 が 0 から離れるほど誤差が大きくなる。これを避けるため関数 $f(x)$ を原点にずらして解く。Picard の逐次近似法と同じように式 (27) のような $f(x)$ を原点に平行移動した関数 $v(x)$ を定義すると、Taylor 展開における Netwon 法を使った逆関数の式は

$$u(x) = \frac{1}{v'(x)} = \frac{1}{f_1 + 2f_2 x + \cdots + n f_n x^{n-1}} \tag{40a}$$

$$q(x) = v(x)\, u(x) \tag{40b}$$

$$g^{\langle 1\rangle} = \frac{1}{f_1}x,$$
$$g^{\langle i+1\rangle} = g^{\langle i\rangle} - q(g^{\langle i\rangle}) + x\, u(g^{\langle i\rangle}) \quad (i = 1, \cdots, L) \tag{40c}$$

$$g(x) = g^{\langle L+1\rangle} + f_c \tag{40d}$$

となる。Newton 法の誤差は二乗の速度で収束するので、n 次まで求めるには

$$L = \lfloor \log_2(n) \rfloor \tag{41}$$

となる。$\lfloor\ \rfloor$ は実数値に対するその値以下の最大の整数値として定義される床関数である。

$g^{\langle L\rangle}$ において n 次まで計算するには、式 (40c) の合成関数計算 $q(g^{\langle i\rangle})$ で n 次、$u(g^{\langle i\rangle})$ で $n-1$ 次まで求める必要がある。$g^{\langle i\rangle}$ は $f(x)$ を原点にずらして解くことから 0 次の係数は 0 である。よって合成関数計算で必要な次数まで厳密に求めるために $q(x)$ は n 次、$u(x)$ は $n-1$ まで求める必要がある。$v(x)$ は $f(x)$ を原点にずらした関数なので、0 次の係数は 0 であることから

$$v(x) = x\,\{r(x)\} \tag{42}$$

とする関数 $r(x)$ を考えることができる。この式 (42) から式 (40b) は

$$q(x) = x\,\{r(x)\, u(x)\} \tag{43}$$

となる。よって $q(x)$ を n 次まで求めるには $n-1$ 次までの $r(x)$ と $u(x)$ が必要となる。$f(x)$ が n 次で与えら

$$
\begin{array}{l|llll}
\left(g^{\langle i+1\rangle}\right)_1 & \left(p^{\langle i\rangle}\right)_0 & & & \\
\left(g^{\langle i+1\rangle}\right)_2 & \left(p^{\langle i\rangle}\right)_1 & \left(\left(p^{\langle i\rangle}\right)^2\right)_0 & & \\
\vdots & \vdots & \vdots & \ddots & \\
\left(g^{\langle i+1\rangle}\right)_{2^i-1} & \left(p^{\langle i\rangle}\right)_{2^i-2} & \left(\left(p^{\langle i\rangle}\right)^2\right)_{2^i-3} & \cdots & \left(\left(p^{\langle i\rangle}\right)^{2^i-1}\right)_0 \\
\left(g^{\langle i+1\rangle}\right)_{2^i} & \underline{\left(p^{\langle i\rangle}\right)_{2^i-1}} & \left(\left(p^{\langle i\rangle}\right)^2\right)_{2^i-2} & \cdots & \left(\left(p^{\langle i\rangle}\right)^{2^i-1}\right)_1 \quad \left(\left(p^{\langle i\rangle}\right)^{2^i}\right)_0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \quad \ddots \\
\left(g^{\langle i+1\rangle}\right)_{2^{i+1}-2} & \underline{\left(p^{\langle i\rangle}\right)_{2^{i+1}-3}} & \underline{\left(\left(p^{\langle i\rangle}\right)^2\right)_{2^{i+1}-4}} & \cdots & \underline{\left(\left(p^{\langle i\rangle}\right)^{2^i-1}\right)_{2^i-1}} \quad \left(\left(p^{\langle i\rangle}\right)^{2^i}\right)_{2^i-2} \quad \ddots \\
\left(g^{\langle i+1\rangle}\right)_{2^{i+1}-1} & \underline{\left(p^{\langle i\rangle}\right)_{2^{i+1}-2}} & \underline{\left(\left(p^{\langle i\rangle}\right)^2\right)_{2^{i+1}-3}} & \cdots & \underline{\left(\left(p^{\langle i\rangle}\right)^{2^i-1}\right)_{2^i}} \quad \underline{\left(\left(p^{\langle i\rangle}\right)^{2^i}\right)_{2^i-1}} \cdots \left(\left(p^{\langle i\rangle}\right)^{2^{i+1}-1}\right)_0
\end{array}
$$

図 3: 形式的 Newton 法における $g^{\langle k\rangle}$ と $p^{\langle k-1\rangle}$ の関係

れることから $v(x)$ が n 次まで求まるので $r(x)$ はこの条件を満たしている。これらから $u(x)$ は $n-1$ 次までの計算となる。

$g^{\langle i\rangle}$ の 0 次の係数は 0 なので、

$$g^{\langle i\rangle} = x\, p^{\langle i\rangle} \tag{44}$$

とする関数 $p^{\langle i\rangle}$ を考えることができる。これから式 (40c) は

$$
\begin{aligned}
& g^{\langle 1\rangle} = \frac{1}{f_1}x, \quad p^{\langle i\rangle} = \frac{g^{\langle i\rangle}}{x}, \\
& g^{\langle i+1\rangle} = g^{\langle i\rangle} - q(x\,p^{\langle i\rangle}) + x\, u(x\,p^{\langle i\rangle}) \\
& \qquad\qquad (i = 1, \cdots, L)
\end{aligned} \tag{45}
$$

となる。

式 (45) の $g^{\langle i+1\rangle}$ を求めるときに $q(x\,p^{\langle i\rangle})$ の合成関数計算で必要とされる $p^{\langle i\rangle}$ のべき乗の係数を $g^{\langle i+1\rangle}$ の各次係数ごとに書き並べると図 3 のような関係となる。$u(x\,p^{\langle i\rangle})$ は x との積となるため、$g^{\langle i+1\rangle}$ から見ると $u(x\,p^{\langle i\rangle})$ の合成関数計算は $q(x\,p^{\langle i\rangle})$ においての一次低い $p^{\langle i\rangle}$ のべき乗計算の値を使った計算になる。よって式 (45) の一回の反復計算中に行われる $q(x\,p^{\langle i\rangle})$ と $u(x\,p^{\langle i\rangle})$ の合成関数計算内でのべき乗部計算は一回で済む。

式 (45) の合成関数の計算は Picard の逐次近似法と同じように、べき乗部の計算値を保持することで計算量の削減ができる。しかし、$q(x\,p^{\langle i\rangle})$ のべき乗部を考えると、図 3 の下線に示される部分は、次ステップの $g^{\langle i+2\rangle}$ において再計算が必要である。$g^{\langle i+1\rangle}$ の 2^i の以降の係数はその反復計算で厳密に求まる値であり、前ステップの $g^{\langle i\rangle}$ でこの係数は厳密に求まっていない。このため前ステップの $g^{\langle i\rangle}$ を使って求める $p^{\langle i\rangle}$ の 2^i-1 以降の係数は、次ステップでの $p^{\langle i+1\rangle}$ の値とは変わってしまう。よって $p^{\langle i\rangle}$ の 2^i-1 以降の係数を使って計算を行う下線部の値は次ステップでは再計算することとなる。

この再計算を避けるため、式 (45) を一回の反復で一つ高次の係数まで求められる反復式と考え、$g^{\langle i\rangle}$ における厳密に求まっていない係数を常に 0 とすることで式 (45) は

$$
\begin{aligned}
& \hat{g}^{\langle 1\rangle} = \frac{1}{f_1}x, \quad \hat{p}^{\langle j\rangle} = \frac{\hat{g}^{\langle j\rangle}}{x}, \\
& \hat{g}^{\langle j+1\rangle} = \hat{g}^{\langle j\rangle} - q(x\,\hat{p}^{\langle j\rangle}) + x\, u(x\,\hat{p}^{\langle j\rangle}) \\
& \qquad\qquad (j = 1, \cdots, \hat{L})
\end{aligned} \tag{46}
$$

となる。この反復計算は一回の反復で一つ高次の係数まで求められ、反復計算で新たに求まる係数より低次の係数は前ステップと同一であることから

$$
\begin{aligned}
& \left(\hat{g}^{\langle 1\rangle}\right)_1 = \frac{1}{f_1}, \quad \hat{p}^{\langle j\rangle} = \frac{\hat{g}^{\langle j\rangle}}{x}, \\
& \left(\hat{g}^{\langle j+1\rangle}\right)_k = \left(\hat{g}^{\langle j\rangle}\right)_k \qquad (k = 1, \cdots, j), \\
& \left(\hat{g}^{\langle j+1\rangle}\right)_{j+1} = \left(\hat{g}^{\langle j\rangle}\right)_{j+1} - \left(q(x\,\hat{p}^{\langle j\rangle})\right)_{j+1} + \left(x\, u(x\,\hat{p}^{\langle j\rangle})\right)_{j+1} \\
& \qquad = -\sum_{k=1}^{j+1} q_k \left(\left(\hat{p}^{\langle j\rangle}\right)^k\right)_{j-k+1} + \sum_{k=1}^{j} u_k \left(\left(\hat{p}^{\langle j\rangle}\right)^k\right)_{j-k} \\
& \qquad\qquad (j = 1, \cdots, \hat{L})
\end{aligned} \tag{47}
$$

となる。

この反復計算を用いて $\lfloor \frac{n}{2} \rfloor$ 次まで求めるため、

$$\hat{L} = \lfloor \frac{n}{2} \rfloor - 1 \tag{48}$$

として、$\hat{g}^{\langle \lfloor \frac{n}{2} \rfloor \rangle}$ まで繰り返す。$\hat{g}^{\langle \lfloor \frac{n}{2} \rfloor \rangle}$ を初期値として式 (45) の反復計算を一回行うことで逆関数を目的の次数まで求めることができる。

$$g^{\langle 1\rangle} = \hat{g}^{\langle \lfloor \frac{n}{2} \rfloor \rangle}$$
$$g^{\langle 2\rangle} = g^{\langle 1\rangle} - q(g^{\langle 1\rangle}) + x\, u(g^{\langle 1\rangle}) \qquad (49)$$

式 (47) は、再計算が発生しないと仮定した式 (45) と比べると若干計算コストが高いが、再計算が発生した式 (45) と比べると計算コストは低い。式 (47) の合成関数のべき乗計算部は式 (45) の合成関数のべき乗計算部で使いまわすことができ、この二つの反復計算を組み合わせることで再計算を避け、効率的に計算することができる。

4　比較実験

これまでの最適化を行った三つの方法とほぼ公式通りの Picard の逐次近似法について 2 次から 40 次までの計算時間を計ったものを図 4 に示す。計算時間は 20 回行った実験結果の平均値である。Lagrange inversion formula の合成関数計算は測定次数から乗算による方法を使っている。実験の入力値には逆正弦関数を使い、正弦関数を求めている。逆正弦関数は奇関数であるがその性質は利用しておらず、入力値を乱数値にした場合もほぼ同じ結果が出ることを確認している。

最適化した Picard の逐次近似法は、Lagrange inversion formula に比べて 20 次で約 1.8 倍、30 次で約 2.0 倍高速である。形式的 Newton と比較すると 20 次で約 1.4 倍、30 次で約 1.2 倍速く、ほぼ公式通りの Picard の逐次近似法とでは 20 次で約 40 倍、30 次で約 57 倍速い。形式的 Newton 法が波打っているのは、最適化手法によって計算効率が偶数の次数までの計算よりも奇数の次数までの計算の方が若干よくなるからである。

また、Lagrange inversion formula の合成関数でのべき乗による方法も加えて 300 次までの計算時間を計ったものが図 5 である。80 次辺りから若干ではあるが最適化した Picard の逐次近似法よりも形式的 Newton 法の計算効率がよくなっている。

この結果から 20 次から 40 次までの範囲では最適化した Picard の逐次近似法が最も優れている。次に低次の場合は Lagrange inversion formula、高次の場合は形式的 Newton 法が優れている。

また FFT を使わない非常に高次での計算では形式的 Newton 法が最も優れているという結果が得られた。

この 20 次から 30 次までの計算効率の順位は他の計算機を使ったとしても変わらないと考えられる。最適化した Picard の逐次近似法は他の方法に比べ、計算結果の使い回しをできる箇所が多く、そのため計算回数が少なくなり、計算効率が上がったものと考えられる。非常に高次での計算で形式的 Newton 法が最も優れていたのは、計算機のキャッシュによる効果ではないかと推測している。

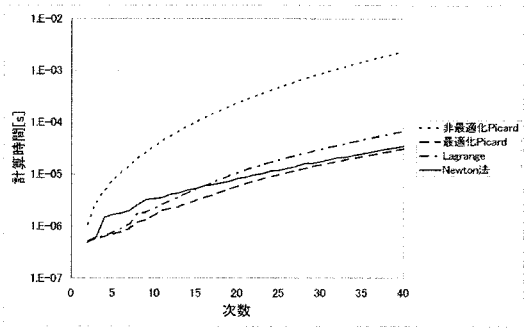

図 4: 逆関数計算の性能比較 1

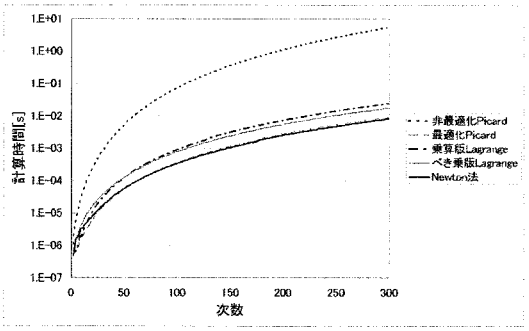

図 5: 逆関数計算の性能比較 2

5　逆関数の応用例

5.1　Gamma 関数の極値

逆関数を含んだ計算の例として Gamma 関数の極大値及び極小値を計算する。Gamma 関数の計算式として式 (50) を使う。

$$\Gamma(x) = \frac{\Gamma(N+x)}{\prod_{k=0}^{N-1}(k+x)} \qquad (50)$$

式 (50) の分子は Stirling の公式 [4] を利用して

$$\log\Gamma(x) \approx \left(x-\frac{1}{2}\right)\log x - x + \frac{1}{2}\log(2\pi) + \sum_{m=1}^{\infty}\frac{B_{2m}}{2m(2m-1)x^{2m-1}} \qquad (51)$$

から計算する。

Gamma 関数の極大値及び極小値を計算するためにまず極値の位置を計算する。極値の位置は微分した Gamma 関数の零点の位置を計算することで求める。

$$\Gamma'(x) = 0 \qquad (52)$$

この方程式は微分した Gamma 関数の逆関数を求め、その逆関数に零を代入することで極値の位置を計算する。

$$x = \Gamma'^{-1}(0) \qquad (53)$$

求まった極値の位置から極大値及び極小値を計算することができる。

Taylor 展開で計算する場合、まず式 (50) の x に $x+a$ を代入して a を中心とした Taylor 級数を求める。初期値を $a=1.5$ とし、$N=20$ で 19 次まで倍精度浮動小数点を係数に使った Taylor 級数で計算し、4 次まで表示させると Gamma 関数は

$$\begin{aligned} 0.8862 \quad & +0.03234(x-1.5) \\ & +0.4148(x-1.5)^2 - 0.1073(x-1.5)^3 \\ & +0.1446(x-1.5)^4 \cdots \end{aligned} \tag{54}$$

となり、これを微分すると

$$\begin{aligned} 0.03234 \quad & +0.8296(x-1.5) \\ & -0.3219(x-1.5)^2 + 0.5786(x-1.5)^3 \\ & -0.3876(x-1.5)^4 \cdots \end{aligned} \tag{55}$$

となる。これから Gamma 関数を微分した式の逆関数を計算すると

$$\begin{aligned} 1.5 \quad + \quad & 1.205(x-0.03234) \\ & +0.5637(x-0.03234)^2 \\ & -0.6941(x-0.03234)^3 \\ & -1.253(x-0.03234)^4 \cdots \end{aligned} \tag{56}$$

となる。この式に $x=0$ を代入し、零点を求めると

$$x = 1.461632144968362 \tag{57}$$

が得られる。よって極値の位置を求めることができる。これを式 (50) に代入することで

$$\Gamma(1.461632144968362) = 0.8856031944108 \tag{58}$$

となる。求まった極値の位置を初期値として反復計算することで任意の精度で極値を求めることができる。また初期値を変えて計算することで任意の位置での極値を求めることができる。これを誤差が 10^{-15} 以下になるまで反復計算した。初期値を変えて計算した結果を表 1 に示す。多倍長精度計算ライブラリを使い十分な精度での計算結果との比較すると、表 1 に示した桁はすべて一致している。

6 おわりに

Taylor 級数における逆関数計算法を三つ挙げ、それらの最適化について述べた。それらとほぼ公式通りの Picard の逐次近似法について計算効率の比較を行った。この結果から倍精度浮動小数点での数値計算における Taylor 級数が与えられた場合の逆関数計算では、最適化を行った Picard の逐次近似法が最も優れている。

Taylor 級数の逆関数計算は微分を含んだ式に対して有用であり、零点や極大極小値、変曲点などを容易に求めることができる。

表 1: Gamma 関数の極値とその位置

初期値	極値の位置	極大極小値	反復回数
1.5	1.461632144	8.856031944e-01	1
-0.5	-0.504083008	-3.544643611e+00	1
-1.5	-1.573498473	2.302407258e+00	2
-2.5	-2.610720868	-8.881363584e-01	2
-3.5	-3.635293366	2.451275398e-01	2
-4.5	-4.653237761	-5.277963958e-02	2
-5.5	-5.667162441	9.324594482e-03	2
-6.5	-6.678418213	-1.397396608e-03	2
-7.5	-7.687788325	1.818784449e-04	2
-8.5	-8.695764163	-2.092529044e-05	2
-9.5	-9.702672540	2.157416104e-06	2

参考文献

[1] Erwin Kreyszig, "Advanced Engineering Mathematics", John Wiley & Sons Inc, 1983

[2] N. De Bruijn, "Asymptotic Methods in Analysis", Dover Pubns, 1981

[3] Louis B. Rall, "Automatic Differentiation: Techniques and Applications", Springer-Verlag, 1981

[4] Milton Abramowitz and Irene A.Stegun, "Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables", Dover Pubns, 1972

[5] 平山 弘 and 小宮 聖司 and 佐藤 創太郎, "Taylor 級数法による常微分方程式の解法", 日本応用数理, Vol.12, No.1, pp.1-8, 2002